UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA


# AUDITORIA CONTÍNUA E OS INCIDENTES DE SEGURANÇA

Rui Pedro Cascalheira Calado

**Mestrado em Segurança Informática**

Versão Pública

Trabalho de Projeto orientado por:
Prof. Doutora Ibéria Vitória de Sousa Medeiros
e Mestre Artur Miguel Adriano Martins

2018

# Agradecimentos

Depois de tantos anos passados na Faculdade de Ciências da Universidade de Lisboa, fecha-se finalmente o ciclo. Fica uma nostalgia pelas histórias e desafios que enfrentei nestes anos e por isso, e tal como comecei, vou dar as graças.

Quero agradecer à carta da FCUL por ter vindo com a palavra "Admitido", quando foi a única opção escolhida e quando não havia um plano alternativo.

Quero agradecer a todas as viagens stressantes e cansativas que fazem parte da vida de um trabalhador-estudante que, depois de estar na faculdade o dia inteiro, segue para o emprego e ao chegar a casa, muitas vezes, tem código para desenvolver.

Obrigado aos SASUL por me terem atribuído bolsas de estudo que possibilitaram os meus estudos e ter alojamento em Lisboa, o que, sem essa ajuda teria sido impossível. E,por me obrigarem a ter uma percentagem mínima de cadeiras feitas por ano para ter acesso a essas bolsas. Sem essa pressão o estudo muitas vezes não teria sido produtivo...

Aos meus colegas e amigos que entraram na faculdade comigo e com quem passei momentos marcantes. Entre as cabeçadas de sono nas aulas teóricas, aos *bugs* nas vésperas das entregas, às noites passadas a programar, ao sofrimento que era ler slides de matéria que nunca mais terminavam...

Quero agradecer à minha orientadora Ibéria Medeiros pela disponibilidade e paciência em ajudar-me a concluir esta dissertação.

Um obrigado à LAYER8, em especial ao Artur Martins, por me ter ajudado a completar esta fase e me apoiar nos primeiros passos desta nova etapa da minha vida.

E como o melhor fica para o fim, um enorme obrigado a ti, Susana, por todo o teu apoio, pela tua gigante paciência e alegria com que me brindas todos os dias.

*À vida.*

# Resumo

A Internet criou oportunidades para as organizações expandirem o seu negócio e partilharem informação, num número crescente de aplicações *web* para aceder a dados criar conteúdos. Tanto as aplicações como as infraestruturas de rede, sendo implementadas por seres falíveis (humanos), contêm vulnerabilidades que podem comprometer o sistema. Para se reduzir o risco de segurança durante o tempo de vida do sistema, é necessária a existência de um processo que permita a identificação, a análise, a monitorização e a mitigação devulnerabilidades continuamente, denominado de *auditoria contínua de segurança da informação*.

Na análise de vulnerabilidades, recorre-se a ferramentas automáticas, denominadas de *scanners*, que identificam vulnerabilidades conhecidas oferecendo as respetivas informações técnicas e sugestões de mitigação. Estes *scanners* devolvem resultados consoante o que foi detetado, e ao serem monitorizados e geridos pela organização, dão origem a incidentes de segurança.

Num processo de auditoria contínua, as organizações executam *scans* ao longo do ciclo de vida do sistema, agendados em curtos intervalos de tempo, em que por cada execução existe um potencial aumento de número de incidentes de segurança apenas referentes a análise de vulnerabilidades.

Tipicamente os incidentes são analisados manualmente, a fim de se tomarem as medidas de resolução das vulnerabilidades encontradas, nomeadamente validar a sua prioridade e criticidade de mitigação. Essa avaliação não só é morosa como tem custos associados, pelos recursos que estão dedicados a esta tarefa. As vulnerabilidades ao serem descritas apenas com informações técnicas, não permitem que seja percetível a prioridade e importância do seu impacto no negócio.

Este trabalho serve para apresentar uma solução que permite minimizar o esforço manual e agilizar a caracterização dos resultados dos *scanner*s em 25%. Através de critérios de negócio e ambiente inerentes ao sistema em análise, é possível calcular um valor de severidade mais realista e gerar alertas que permitam prioritizar a mitigação das vulnerabilidades encontradas.

**Palavras-chave**: segurança da informação, auditoria contínua, análise de vulnerabilidades, severidade, alertas.

# Abstract

The Internet has created opportunities for organizations to expand their businesses and share information, supported by a growing number of web applications to access data and create content. Both applications and network infrastructures, being implemented by fallible beings (humans), may have vulnerabilities that can compromise the whole system. In order to reduce the security risk throughout system lifecycle, there is a need for a process that allows the continuous identification, the analysis, the monitoring and vulnerabilities mitigation. Such a process is named *continuous auditing of information security*.

Vulnerability Assessments uses automatic tools, called *scan*ners, that identify known vulnerabilities by providing their related technical information and mitigation suggestions. These *scanners* show the results based on what has been detected, and after being monitored and managed by the organization, originate new security incidents.

In a continuous auditing process, organizations run *scans* over system life, scheduled in short time periods, and for each execution there is a potential increase in the number of security incidents related to the vulnerability assessment only.

Typically, the incidents are analyzed manually, in order to take measures regarding the findings, namely to validate their priority and criticality of mitigation. This evaluation is not just time-consuming but also costly due to the resources that are allocated to it. Vulnerabilities, since being described with just technical information, don't allow the perception of their priority and the importance of their impact on the business.

This thesis presents a solution that will contribute to minimize the manual effort and to speed up the characterization of the results of the *scanners* by 25%. Through business and environmental criteria inherent to the system under analysis, it is possible to calculate a more realistic risk score and to generate alerts that allow prioritizing the mitigation of the vulnerabilities.

**Keywords**: information security, continuous auditing, vulnerability assessment, risk score, alerts

# Conteúdo

# Lista de Figuras

# Lista de Tabelas

# Capítulo 1

# Introdução

Nos primórdios da Internet, a rede de computadores existente servia essencialmente para partilha de documentos e conhecimento entre uma pequena comunidade científica [1], cenário que se alterou. A rede evoluiu para uma escala planetária, possibilitando comunicações e trocas de informação quase instantâneas, que influenciam não só a vida social como o mundo empresarial. Fruto disso, o volume de informação / dados na Internet tem aumentado exponencialmente, impulsionado pelo crescente número de aplicações *web* que são a forma mais utilizada para aceder a dados e criar conteúdos Esse aumento é de tal maneira acentuado que, em dois anos, será produzida mais informação do que em toda a história da Humanidade e pelo valor que a informação possa ter, atualmente a privacidade e a sua segurança são os tópicos mais discutidos [2].

Com o crescimento do acesso à Internet, o número de ataques informáticos sofreu também um aumento, efeito de diversos fatores: quantidade de possíveis vítimas que estão interligadas, o sentimento de poder e anonimato de quem tem conhecimentos técnicos de segurança e redes de computadores que podem facilitar a inclusão em atividades criminosas, o aproveitamento da falta de conhecimento ou sensibilidade para as questões de segurança (ex.: desenvolvimento de aplicações seguras ou configurações e atualizações do sistema). Embora existam mecanismos de defesa de perímetro, como por exemplo *firewalls*, detetores de intrusão e comunicações seguras, é difícil mudar o foco da gestão das organizações para o problema e preparar os sistemas para possíveis ataques informáticos [3] que os tornem inacessíveis (ataque de negação de serviço) ou permitam acesso indevido a dados por parte de utilizadores ilegítimos.

A informação cobiçada passa não só por informações governamentais e tecnológicas mas também dados pessoais (como dados bancários ou clínicos) [4]. Indivíduos maliciosos com esses conteúdos em mãos podem vendê-los no mercado negro (por elevadas quantias de dinheiro) a outros governos ou instituições para tomada de ações políticas, venda do historial médico de pacientes a seguradoras, farmacêuticas ou bancos que podem usá-las para aumentar, por exemplo, o custo dos seus serviços.

As organizações devem implementar medidas de segurança, por camadas, a fim de possuírem uma estrutura e processos bem definidos. No entanto, deverá estar sempre presente a ideia de que um sistema não é infalível e não é livre de falhas de configuração ou erros em *hardware* ou *software*, que um agente malicioso possa explorar. Aplicações ao serem implementadas por seres falíveis (humanos), podem conter erros, designados de *bugs*, no código fonte, os quais se devem a más práticas de programação, falta de conhecimento de como implementar código seguro, ou descoro da segurança para lançar mais rapidamente o produto para o mercado. Estes *bugs*, se envolverem dados de fontes externas (ex. *inputs*, ficheiros), representam uma forma de abusar de fraquezas do sistema, e denominam-se de *vulnerabilidades* que têm o potencial de comprometer o sistema, se exploradas.

Com o crescimento exponencial de *software* proporcionado pela Internet, tais como aplicações *web* e diferentes sistemas operativos, o leque de vulnerabilidades aumenta cada vez mais. Existe cerca de 1 a 25 *bugs* por cada 1000 linhas de código fonte [5] e dado o tamanho de algum *software* (ex: os sistemas operativos contêm milhões de linhas de código) pode-se inferir que dificilmente um produto será totalmente isento de vulnerabilidades, isto é, 100% seguro. A diversidade de aplicações, o tempo de implementação *vs* lançamento para o mercado, as más práticas de programadores, a falta de requisitos de segurança no desenvolvimento do *software* e a quase ausência de testes de *software* (nomeadamente de segurança) são fatores diretos que contribuem para o crescente aumento de vulnerabilidades. Por exemplo, 55% das aplicações web têm pelo menos uma vulnerabilidade crítica e 84% das vulnerabilidades que contêm são consideradas médias [6]. Entre as vulnerabilidades, as de SQL Injection e Cross Site Scripting são as mais recorrentes, pelo que se posicionam no top três do OWASP Top 10 de 2017 [7].

Assim como o software, também o número de equipamentos ligados entre si aumentou, formando inúmeras redes que comportam milhões de utilizadores, sendo a taxa anual de aumento de máquinas a rondar os 51% nos últimos dois anos e sem previsão de abrandamento. Com mais máquinas, surgem mais serviços e mais utilizadores. O uso e acesso, aliado à quantidade de informação produzida, tornam-se não só uma ameaça de segurança para indivíduos e interesses comerciais, como também para nações [8]. Por isso, pela mutabilidade que as redes atuais devido a dispositivos móveis e à conectividade que se prevê existir na maioria dos componentes eletrónicos (Internet of Things) permitem, é imperativo seguir as boas políticas de segurança para

evitar vulnerabilidades de sistema como versões de componentes e serviços obsoletas, más configurações ou ausência de controlos de segurança [9].

A constante evolução e sofisticação de ataques, cada vez mais inovadores e organizados, ameaça os sistemas mais complexos. Portanto, é crucial que não só a infraestrutura física e rede do sistema esteja protegida como também existam medidas de segurança durante o ciclo de desenvolvimento de software.

Um sistema, não só deve proteger os seus dados como deverá ser alvo de uma análise contínua do nível de segurança de todas as aplicações nele inseridas, criando um processo que identifique, analise, mitigue e reporte vulnerabilidades de forma a reduzir a ameaça de ataques informáticos.

## 1.1. Motivação

A Internet abriu uma janela de oportunidades para as organizações expandirem o seu negócio e partilharem informação. No entanto, também levantou questões de segurança associada à exposição da informação, aplicações e infraestruturas críticas e dados pessoais. Torna-se impreterível proteger adequadamente a informação seguindo uma abordagem estruturada, bem avaliada e compreensível para se reduzirem os riscos de segurança.

Pela mutabilidade dos sistemas, as entradas e saídas de equipamentos ou alterações em aplicações, a abordagem tradicional de realizar testes de segurança pontualmente deixa de fazer sentido. Tal necessidade é reforçada quando, segundo a base de dados de CVEs, o ano de 2017 contou com cerca de 12500 novas vulnerabilidades em que 10% são críticas, num crescimento de 51% face ao ano anterior [10]. Todos estes factos, associados à conformidade exigida pelos reguladores, justificam a necessidade de encarar os vários testes de segurança num âmbito de continuidade e assim obter uma visão atualizada da segurança da organização. Dois dos testes de segurança que vêm sendo aplicados esporadicamente são os testes de intrusão e a análise de vulnerabilidades.

Os testes de intrusão têm o intuito de avaliar a dificuldade de um atacante em contornar os controlos de segurança implementados, recorrendo a ferramentas manuais ou automáticas, a fim de conseguir o acesso indevido à informação e ao sistema.

A análise de vulnerabilidades trata-se do processo de varrimento (i.e., *scan*) de um software ou rede de forma a encontrar falhas e fraquezas (vulnerabilidades), sejam elas conhecidas até então ou não, no sistema. Trata-se de um tipo de teste que utiliza

ferramentas automáticas, denominadas *scan*ners, que devolve inúmeros resultados, consoante as vulnerabilidades identificadas e informações do alvo tais como serviços disponíveis, aplicações ou sistema operativo. A análise de vulnerabilidades é uma das primeiras ações para avaliar o nível de segurança da organização e normalmente precede todos os outros tipos de teste de segurança [11].

Estes testes de segurança mencionados, são uma mais valia que oferecem um snapshot do nível de segurança do sistema num dado momento. No entanto, por não haver acompanhamento contínuo dos resultados, esses relatórios tornam-se obsoletos à medida que novas vulnerabilidades vão surgindo. Por isso, é estritamente importante que exista uma componente contínua de segurança capaz de identificar as vulnerabilidades ao longo do tempo e, assim, testemunhar o nível de segurança atual do sistema.

Após cada *scan*, como referido anteriormente, são apresentados resultados que identificam as vulnerabilidades encontradas. Por cada resultado, é desencadeado um pedido de análise de incidente de segurança. Num processo de *auditoria contínua*, as organizações executam *scans* ao longo do ciclo de vida do sistema, agendados em curtos intervalos de tempo, em que por cada execução existe um grande potencial de aumento de número de incidentes apenas referentes a análise de vulnerabilidades. Tipicamente estes incidentes de segurança são analisados manualmente, a fim de se tomarem as medidas de resolução das vulnerabilidades encontradas, nomeadamente a validação da sua prioridade e criticidade de mitigação. Essa avaliação não só é morosa como tem custos associados, pelos recursos que estão dedicados a esta tarefa. As vulnerabilidades, ao serem descritas apenas com informações técnicas, não permitem que seja percetível a prioridade e importância do seu impacto no negócio.

Sendo a LAYER8 uma empresa especializada em segurança da informação, esta pretende oferecer uma plataforma de auditoria contínua de segurança informática que permita agilizar e minimizar o esforço da caracterização das vulnerabilidades encontradas. Através de critérios de negócio e ambiente inerentes ao sistema em análise, é possível calcular um valor de severidade mais realista e gerar alertas que permitam um nível de automação à caracterização das vulnerabilidades encontradas.

## 1.2. Objetivos

Pretende-se criar uma metodologia de cálculo do risco de vulnerabilidades que devolve *scores* de severidade (parciais e final). Através do estudo de metodologias e métricas já utilizadas, é proposta uma nova metodologia que oferece métricas não

contempladas até agora, mais direcionadas para o negócio e ambiente do sistema analisado.

É definido um gerador de alertas, que com base nos resultados da metodologia criada, emite diferentes alertas.

Por fim, é realizada a implementação do gerador e da metodologia de cálculo do risco, obtendo uma calculadora e um gestor de alertas. A implementação é então avaliada a partir de cenários reais em sistemas de diferentes contextos.

## 1.3. Contribuições

- Estudo de modelos e calculadoras de cálculo de severidade de vulnerabilidades já existentes e a consequente identificação de métricas de negócio e ambiente que complementem e se revelem mais valias;
- Criação e implementação de uma calculadora de *score* de severidades de vulnerabilidades mais realista, de acordo com o sistema analisado;
- Desenvolvimento de um gerador de alertas capaz de reduzir o esforço de análise da equipa de resposta a incidentes e que auxilie na organização de incidentes de segurança.

## 1.4. Organização do documento

Este documento está organizado da seguinte forma:

No capítulo 2 é apresentado o contexto do ciclo de desenvolvimento de software integrando a componente de segurança no software. O conceito de vulnerabilidade de software e diferentes classes de vulnerabilidades serão estudadas bem como os tipos de testes de segurança que se podem realizar para as descobrir/prevenir/explorar, com maior foco em análise de vulnerabilidades. É realizada uma apreciação sobre vários modelos de gestão de risco de ameaças e cálculo do nível de ameaça.

No capítulo 3 vai-se incidir na metodologia de cálculo de risco criada e calculadora resultante, bem como no desenvolvimento do gerador de alertas para um processo de auditoria contínua de segurança de informação.

O capítulo 4 trata da avaliação tanto da calculadora como do gerador de alertas utilizando cenários reais em que a LAYER8 exerceu atividade.

No capítulo 5 são tiradas as devidas ilações sobre o trabalho desenvolvido e quais as perspectivas de futuro para que solução evolua e responda inteiramente ao problema identificado.

# Capítulo 2

# Contexto e trabalho relacionado

A segurança da informação é uma área que tem vido a ganhar notoriedade pela necessidade de preservar os dados de algo ou alguém, sendo que o risco de ataques informáticos em larga escala tem vindo a aumentar devido à sofisticação dos ataques e à híper-conectividade, crescimento de dispositivos com acesso à Internet e dados sensíveis (ex.: financeiros e médicos) guardados na *cloud* [12]. Por se tratar de um domínio científico, existem terminologias que devem ser aprendidas ou revistas.

Neste capítulo irão abordar-se os conceitos elementares e as propriedades da segurança da informação. De seguida, caracteriza-se o desenvolvimento de software e como este é influenciado pela segurança, de forma a produzir software seguro. Indicam-se os testes de segurança tipicamente utilizados na identificação de vulnerabilidades bem como é feita a avaliação do risco no que diz respeito à caracterização das vulnerabilidades. Por fim, é apresentado em que consiste uma auditoria contínua de segurança, o que é e como se caracterizam incidentes de segurança.

## 2.1. Conceitos e propriedades de segurança

A segurança da informação assenta em três propriedades fundamentais: *confidencialidade, integridade* e *disponibilidade*. A confidencialidade é a propriedade que garante a ausência de divulgação não autorizada de informação. A integridade trata de assegurar a não alteração não autorizada do sistema ou informação, e a disponibilidade garante a prontidão do sistema para fornecer um serviço ou informação. A designação “autorizada” é dada pelo conjunto de requisitos de segurança satisfeitos pelo sistema, isto é, permissões.

Um dos conceitos chave da segurança da informação é a vulnerabilidade. Uma vulnerabilidade é definida como uma falha no projeto ou implementação de *software* ou processo que, quando explorada por um atacante, resulta no comprometimento de um computador ou sistema. As vulnerabilidades podem ser classificadas em três categorias:

- Vulnerabilidade de desenho ou projeto – este tipo de vulnerabilidade é introduzido durante o levantamento de requisitos e desenho do sistema. Exemplos deste tipo de vulnerabilidade é a decisão de utilizar um mecanismo de autenticação fraco ou não considerar que um atacante pode escutar a comunicação entre um cliente e o servidor;
- Vulnerabilidade de implementação ou codificação – introduzidas durante a implementação do sistema, ou seja, durante o desenvolvimento do código do sistema. Usualmente são *bugs* de programação criados pelos programadores não intencionalmente, mas com implicações ao nível da segurança ou uma discrepância entre os requisitos definidos e o código desenvolvido. Um exemplo é a não utilização de *queries* parametrizadas ao interagir com uma base de dados;
- Vulnerabilidade de gestão ou operacionais – são vulnerabilidades causadas pela gestão do ambiente no qual o sistema é executado ou pela sua configuração. Um exemplo é a existência de contas com palavras-passe por defeito num sistema de gestão de bases de dados.

As vulnerabilidades podem ou não ser exploradas por *inputs* externos ao sistema, provindos de qualquer pessoa mal-intencionada. Se o forem, essa ação maliciosa designa-se pelo nome de ataque e a interface por onde o ataque é feito, isto é, a área exposta a um ataque, designa-se superfície de ataque. A superfície de ataque é constituída por interface do utilizador, rede, interface para uma terceira parte (*software*), sistema de ficheiros, sistema operativo ou aplicação alvo. A técnica pela qual o atacante consegue ganhar acesso indevido de uma rede ou equipamento chama-se de vetor de ataque. Exemplos de vetor de ataque são vírus, páginas *web*, anexos de *email*, etc.

Em ataque, o agente malicioso pode utilizar *scripts*, dados ou sequência de comandos para explorar uma determinada vulnerabilidade de um *software*, conhecido por *exploits*. Para mitigar ou prevenir vulnerabilidades de *software*, são publicados *patchs* (excerto de *software* que serve para eliminar vulnerabilidades e outros *bugs*, ou melhorar a usabilidade ou desempenho do produto) pelos proprietários do produto. Caso um ataque seja bem-sucedido, existe uma intrusão que poderá suscitar um erro e o mesmo levar a uma falha do sistema. Sucintamente, a exploração de uma

vulnerabilidade num sistema é representada pelo modelo AVI (ataque, vulnerabilidade e intrusão), através da expressão

| Vulnerabilidade + Ataque → Intrusão |
|---|

A Figura 2.1 ilustra o referido modelo.

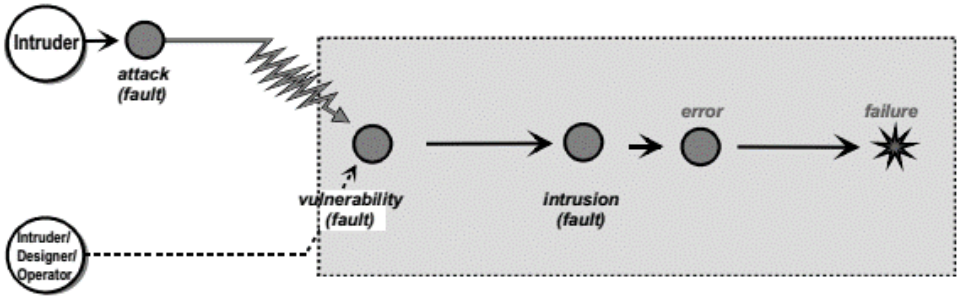


Figura 2.1: Modelo AVI [13]

Como já foi referido, o facto de existir uma intrusão não significa que o sistema esteja comprometido, porque irá depender não só do conhecimento, perspicácia e experiência do atacante, como também dos controlos de segurança que o sistema deverá ter. Deve-se assumir sempre que o sistema não é 100% seguro, pelo número de vulnerabilidades que surgem diariamente, pelas novas técnicas de ataque de criminosos, pelo "errar é humano" dos programadores que desenvolvem software e que também cometem erros. É importante perceber que a informação tem valor e por isso os sistemas, consoante as políticas, metodologias e infraestrutura de segurança que implementam, têm maior ou menor risco associado. O risco é o efeito da incerteza num objetivo. Sendo que o objetivo é a segurança da informação, então, está associado ao potencial em que ameaças possam explorar vulnerabilidades num ativo de informação ou grupo de ativos de informação e causar dano à organização [14]. Esta métrica é indispensável pela representação quantitativa que permite condensar a informação recolhida sobre o sistema em algo concreto, palpável. Direciona as preocupações e prioridades na segurança consoante o nível de risco e assim fazer um devido acompanhamento do estado do sistema, planear e racionalizar recursos, com o objetivo do reduzir a níveis considerados aceitáveis. Para tal, dado que o nível de ameaça, mesmo com o combate ao cibercrime, não se consegue quantificar, o fator decisor para reduzir o risco incide no grau de vulnerabilidade através da não existência dos três tipos de vulnerabilidade e na implementação de controlos e processos de segurança.

## 2.2. Desenvolvimento do software seguro

As equipas que produzem software têm cinco aspetos nos quais se focam – *funcionalidade*, *usabilidade*, *desempenho*, *simplicidade* e *time-to-market*.

- Na funcionalidade, pretende-se fornecer um conjunto de tarefas ao utilizador, o que normalmente implica num aumento de complexidade;
- A usabilidade tem em conta a facilidade do utilizador em interagir com o *software*;
- No desempenho, é necessário que a eficiência do produto esteja em concordância com o objetivo do mesmo;
- A simplicidade do produto para que seja mais intuitivo para com os seus utilizadores;
- O baixo *time-to-market* representa o foco na rápida colocação do produto em produção (no mercado) para que se rentabilize o mais breve possível.

Com estes objetivos em mente, as equipas de desenvolvimento seguem um processo que é representado essencialmente por cinco fases: *requisitos*, *desenho*, *implementação*, *testes*, e *operação*, como ilustrado na Figura 2.2.


Figura 2.2: Processo de desenvolvimento de software [15]

Na fase 1, de requisitos, definem-se os requisitos necessários para alcançar os objetivos do projeto, tipicamente de negócio e de eficácia.

Na fase 2, desenho, são detalhadas as funcionalidades do sistema, o que inclui as interfaces com o utilizador, regras de negócio, diagramas de processos (*workflows*), entre outra documentação, que têm como *input* os requisitos aprovados anteriormente.

Na fase 3, implementa-se o desenho criado na fase 2, transformando-o num sistema de informação operacional. São implementados os ambientes, bases de dados, código-fonte e os procedimentos de casos de teste.

Na fase 4, de testes, o código é testado a vários níveis. Os testes podem ir desde testes unitários, de sistema, até testes de aceitação. Existem diversas metodologias de

testes, contudo todas têm como objetivo comum a identificação de *bugs*, correção e validação da correção.

Por fim na fase 5, passagem a produção (operação), o sistema é colocado em produção fazendo parte do negócio da organização, onde os problemas "reais" aparecem devido à interação do utilizador com o produto e que precisam de ser resolvidos.

A segurança informática tem uma influência negativa nos objetivos das equipas de desenvolvimento, apresentada na Tabela 2.1,

| | | |
|---|---|---|
| Segurança VS | Funcionalidade | Para *software* mais seguro, este deverá oferecer menos serviços ou funcionalidades |
| | Usabilidade E Desempenho | Implementação de medidas de segurança não ajudam os utilizadores a concretizar tarefas e prejudicam a ação das aplicações e a gestão de recursos (ex: mecanismos de autenticação e validações de *input*) |
| | Simplicidade | Mecanismos de segurança reduzem os custos de produção e manutenção, consequentemente, a simplicidade |
| | Time-to-market | Testes de segurança prejudicam o lançamento dos produtos no mercado |

Tabela 2.1: Influência da segurança informática em desenvolvimento de software

e nas várias fases de desenvolvimento de *software*, existem potenciais problemas de segurança que podem originar vulnerabilidades, aumentando o risco do *software*. Ao invés de uma postura de *security by design*, isto é, a segurança envolvida no desenvolvimento de software desde a fase de requisitos, a presença da segurança de informação surge já na fase de testes. As organizações têm a tendência de investir em testes de segurança quando o tempo e o orçamento assim o permitem, havendo uma postura de segurança mais reativa e à posteriori da implementação do produto. O resultado que advém destas decisões é um enorme número de vulnerabilidades que, para serem corrigidas, revelam custos e um esforço elevados.

Pela Figura 2.3, verifica-se que o custo por vulnerabilidade encontrada é exponencial ao longo das várias fases.


Figura 2.3: Custo de defeito ao longo do tempo [16]

O *software* nas fases 1 e 2 (requisitos e desenho, respetivamente) está sujeito a vulnerabilidades de projeto, onde as decisões tomadas podem não estar de acordo com a legislação em vigor, recomendações ou normas internacionais da segurança da informação. Nestas fases, ainda não existe software implementado, logo modificações ou correções têm pouco impacto no projeto. A segurança é envolvida definindo casos de uso onde são identificadas interações de possíveis atacantes com base em lista de problemas comuns no tipo de *software*, de recursos que podem ser abusados e nos casos de uso para utilizadores legítimos. O impacto no custo total e nos prazos de implementação do projeto são muito inferiores aos que existiriam caso as vulnerabilidades fossem detetadas à posteriori.

Na fase 3, de implementação, a programação e as configurações dos componentes do sistema devem ser realizadas de modo a não criar vulnerabilidades de implementação. Formas de implementação de código seguro, tais como, validação e sanitização dos *inputs* dos utilizadores, boas práticas de programação, a escolha da linguagem, as definições e configurações dos vários elementos do sistema devem ser ponderados de forma a diminuir o risco do sistema o mais possível.

Na fase 4, de testes, já existe produto e por isso uma vulnerabilidade de segurança encontrada exige comunicação entre quem testa e quem desenvolve para que seja identificada, corrigida e validada. Aqui o fator custo/benefício é muito importante porque nunca é possível realizar todos os testes.

Por fim, na fase 5 (operação), todos os requisitos foram cumpridos e a sua implementação foi validada. Nesta fase, as vulnerabilidades que possam existir são do tipo gestão, que irão surgir caso os requisitos obrigatórios identificados em fases anteriores não tenham sido cumpridos ou bem executados, isto é, acabam por conduzir a

vulnerabilidades de implementação ou de projeto. Uma vulnerabilidade, se encontrada e explorada, vai ter um impacto enorme, não só financeiro, como de reputação. O *software* além dos desafios funcionais que serão colocados por estar a ser utilizado, enfrenta também uma possível ameaça de quem pretende abusar dos seus serviços para comprometer o sistema ou obter informações de forma ilícita. Uma vulnerabilidade identificada e explorada que permita a divulgação de dados de utilizadores, além de influenciar a credibilidade e reputação da organização na proteção informática aos seus clientes, aumenta o nível de ameaça por despertar atenção de pessoas maliciosas que podem focar-se nessa fragilidade para comprometer o sistema. Quando estes incidentes acontecem, alocam-se muitos recursos humanos (que têm de ser pagos) para os corrigir e criar um *patch* para os utilizadores de modo a remover essa vulnerabilidade o mais breve possível. Enquanto esse *patch* não for publicado, a reputação e os dados estão em iminência de serem comprometidos. Mesmo após a sua publicação, como são os utilizadores os responsáveis por instalá-los, potenciam custos de administração e instabilidade dos sistemas para os utilizadores. Por esse motivo, a política de *patchs* de algumas organizações torna-se muito complexa, pouco eficaz ou, por outro lado, não existe.

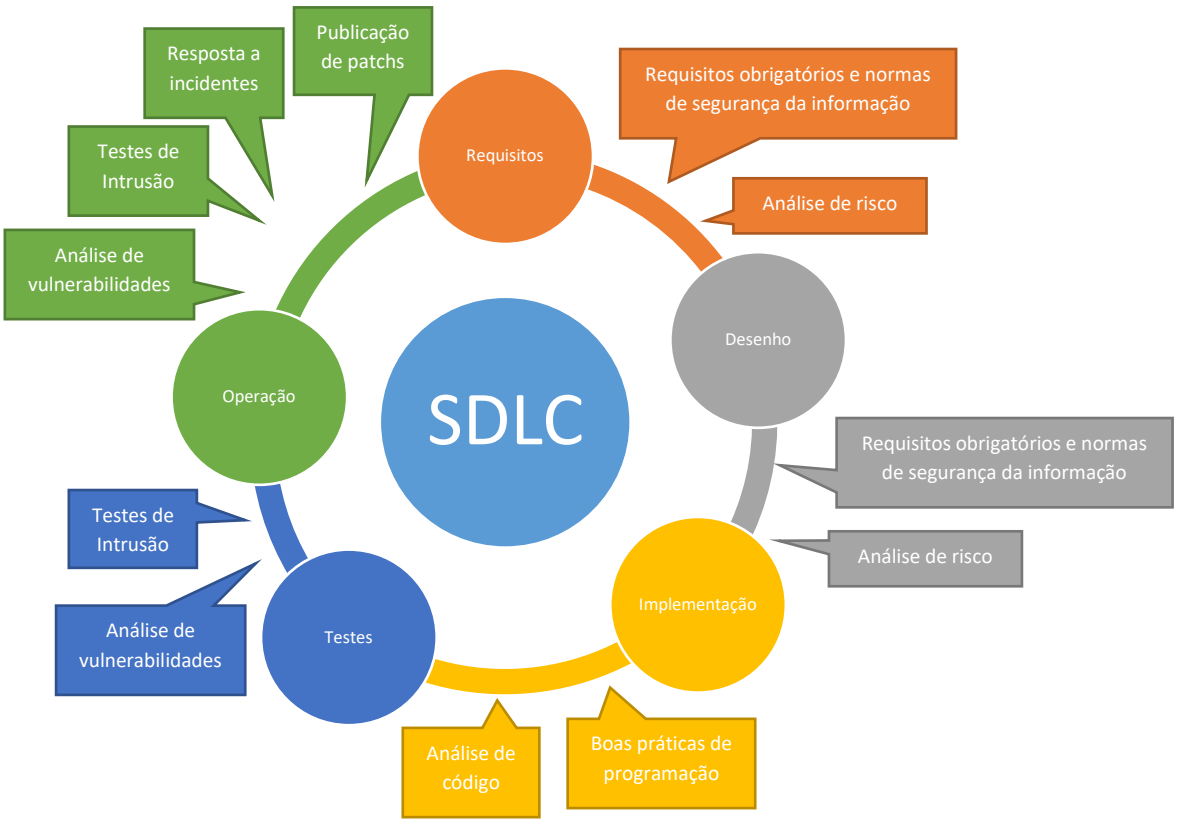


Figura 2.4: Desenvolvimento de software seguro [17]

Pela realidade do aumento de ataques informáticos, e tendo o desenvolvimento de *software* um papel fundamental na redução do risco e aumento da segurança das organizações, a preocupação por uma abordagem *security by design* tem emergido. A inclusão da segurança num processo de desenvolvimento de *software* é crucial para uma atitude proactiva em reduzir o maior número de vulnerabilidades. Assim sendo, existem várias medidas e testes de segurança que podem ser aplicadas em cada fase do ciclo de desenvolvimento do software (SDLC) (Figura 2.4).

Nos requisitos e desenho, é necessária uma avaliação do risco que identifique os requisitos de segurança consoante as normas e leis em vigor quanto à segurança da informação, as ameaças, impacto de um possível ataque informático e quais os alvos mais aliciantes do sistema, isto é, os mais críticos. Existe também o estudo de quais os casos de uso indevidos e o planeamento dos restantes testes de segurança.

Na implementação, realizam-se auditorias de segurança através de análises dinâmicas e estáticas ao código. Para tal, analisa-se linha a linha, função a função, o fluxo dos dados e as validações ao longo do código de modo a identificar vulnerabilidades e ações que possam comprometer o sistema.

Na fase de testes, fazem-se análises de vulnerabilidades no sentido de identificar assinaturas de vulnerabilidades já conhecidas. Otimiza-se essa tarefa através de ferramentas automáticas – *scanners*. São feitos também testes de intrusão, que simulam ações maliciosas de um atacante. É definido um âmbito e uma metodologia no sentido de testar o que é mais crítico mediante as necessidades.

Por fim, na operação, devem ser realizadas análises de vulnerabilidades e testes de intrusão periodicamente, de modo a identificar e corrigir, no menor tempo possível, vulnerabilidades entretanto descobertas. Deve existir também um processo de resposta a incidentes que acompanhe a resolução das vulnerabilidades que poderá culminar na publicação de *patchs*.

**RESTANTE CONTEÚDO REMOVIDO POR MOTIVOS DE CONFIDENCIALIDADE**

# Capítulo 3

# Confidencial

# Capítulo 4

# Confidencial

# Capítulo 5

# Conclusão

Este estudo teve como intenção entender como agilizar e minimizar o esforço da caracterização das vulnerabilidades encontradas de forma automática num processo de auditoria contínua, evitando um esforço de análise manual pela parte da equipa de resposta a incidentes. Foi dada também uma visão de como surgem as vulnerabilidades, o que se pretende numa auditoria contínua, os desafios dos incidentes de segurança e da equipa que os gerem.

Foram estudadas diversas técnicas de identificação e caracterização de vulnerabilidades como análise de vulnerabilidades e testes de intrusão, bem como metologias de cálculo de valor de severidade de vulnerabilidades, para entender qual a melhor forma de abordar este problema em organizações.

Através deste estudo foi possível desenvolver uma calculadora de *score* de severidades de vulnerabilidades baseada em metodologias de cálculo já existentes com novas métricas de negócio e ambiente, que permitem minimizar o esforço de análise dos resultados obtidos pelos *scanners*. A calculadora é composta por três módulos, que combinados, permitem o cálculo da severidade das vulnerabilidades, de forma mais próxima da realidade do sistema. Com os resultados da calculadora, desenvolveu-se um gerador de alertas que permite minimizar o esforço de análise dos incidentes de segurança.

A calculadora e o gerador de alertas foram implementados e avaliados em ambiente de teste, utilizando relatórios de vulnerabilidades anteriormente realizados pela LAYER8.

Os valores obtidospermitiram verificar que os resultados da calculadora foram muito próximos dos obtidos pelos auditores da LAYER8, permitindo demonstrar que foi possível, de forma automática, recriar a análise manual dos relatórios. Quanto ao gerador de alertas, por base na calculadora desenvolvida neste trabalho, existiu uma minimização de 25% no esforço em caracterizar os incidentes de segurança.

# Bibliografia


[1] F. Pinto, Auditoria Contínua: Um novo paradigma de auditoria, Instituto Politécnico do Porto, 2011.

[2] B. Marr, "Big Data: 20 Mind-Boggling Facts Everyone Must Read," Setembro 2015. [Online]. Available: https://www.forbes.com/sites/bernardmarr/2015/09/30/big-data-20-mind-boggling-facts-everyone-must-read/#79557caf17b1.

[3] A. Suspiro, "Ataques informáticos. 23% das empresas portuguesas foram alvo," Outubro 2015. [Online]. Available: http://observador.pt/2015/10/29/ataques-informaticos-23-das-empresas-portuguesas-foram-alvo/.

[4] Gemalto, "2015 Data Breach Statistics: The Good, the Bad and the Ugly," Março 2016. [Online]. Available: https://blog.gemalto.com/security/2016/03/03/2015-data-breaches-by-the-numbers/.

[5] S. C. McConnell, Code Complete, Microsoft Press, 2004, p. 521.

[6] Acunetix, "Web Application Vulnerability Report 2016," Acunetix, 2016.

[7] Stock, T. Gigler and B. Glas and N. Smithline and A. van der, "OWASP Top 10: The Ten Most Critical Web Application Security Risks -- RC2," OWASP Foundation, 2017.

[8] J. M. Kizza, Guide to Computer Network Security, Springer International Publishing, 2017.

[9] K. Beaver, "2015 Data Breach Statistics: The Good, the Bad and the Ugly," 31 Julho 2013. [Online]. Available: https://www.acunetix.com/blog/articles/the-top-5-network-security-vulnerabilities/.

[10] CVE Details, "CVE Details," 13 November 2017. [Online]. Available: https://www.cvedetails.com/.

[11] S. M. Irfan yaqoob, "Penetration Testing and Vulnerability Assessment," *Journal of Network Communications and Emerging Technologies (JNCET),* vol. 7, Agosto 2017.

[12] World Economic Forum, "Global Risks 2015 10th Edition," World Economic Forum, 2015.

[13] N. Neves, J. Antunes, M. Correia, P. Veríssimo e R. Neves, "Using Attack Injection to Discover New Vulnerabilities," 2006.

[14] ISO/IEC 27005, “Information technology — Security techniques — Information security risk management,” 2011.

[15] ISTQB, “What are the Software Development Life Cycle (SDLC) phases?,” 2017.

[16] B. W. Boehm, Software Engineering Economics, Prentice Hall PTR, 1981.

[17] Microsoft, “What is the Security Development Lifecycle ?,” Microsoft, [Online]. Available: https://www.microsoft.com/en-us/sdl. [Acedido em 17 04 2018].

[18] S. R. K. Kranthi Kumar.K, “A Latest Approach to Cyber Security Analysis using Vulnerability Assessment and Penetration Testing,” 2014.

[19] Guru99.

[20] J. Manico, “Secure Development Lifecycle”.

[21] M. M. Morana, “Building Security Into The Software Life Cycle - A Business Case”.

[22] Federal Office for Information Security, “BSI - Study A Penetration Testing Model”.

[23] LAYER8, *Documentação interna - Metodologia de Teste de Intrusão,* 2016.

[24] RedHat, “Security Guide”.

[25] Alienvault, “Vulnerability assessment remediation”.

[26] Risk Based Security, “Network vulnerability assessment”.

[27] Jason Creasey, Ian Glover, “A guide for running an effective Penetration Testing programme,” 2017.

[28] F. Viggiani, *Design and implementation of a non-aggressive automated penetration testing tool,* KTH Royal Institute of Technology, 2013.

[29] Synopsys, “Vulnerability assessment”.

[30] C. T. Phong, *A Study of Penetration Testing Tools and Approaches,* Auckland University of Technology, 2014.

[31] O. C. Bellatriu, *Penetration Testing Automation System,* Barcelona School of Informatic, 2014.

[32] S. Watts, *IT Security Vulnerability vs Threat vs Risk: What’s the Difference?,* BMC, 2017.

[33] T. Bass, *adaptada de The Top Ten Security Threats for 2008 - Risky Situations and Context,* The cyberpace event processing blog, 2017.

[34] TAG, *Threat, vulnerability, risk – commonly mixed up terms,* Threat Analysis Group.

[35] S. Flowerday, “Continuous auditing technologies and models: A discussion,” *Computers and Security,* 2006.

[36] K. V. Impe, “Simplifying Risk Management,” 2017.

[37] S. Elky, “An Introduction to Information System Risk Management,” 2006.

[38] J. Bayne, “An Overview of Threat and Risk Assessment,” 2002.

[39] NIST800-30, “Risk Management Guide for Information Technology Systems,” 2002.

[40] NIST, “Managing Information Security Risk: Organization, Mission, and Information System View,” National Institute of Standards and Technology, 2011.

[41] First, “Common Vulnerability Scoring System SIG,” [Online]. Available: https://www.first.org/cvss/. [Acedido em 05 2018].

[42] Mitre, “Common Weakness Enumeration,” [Online]. Available: https://cwe.mitre.org/cwss/cwss_v1.0.1.html. [Acedido em 05 2018].

[43] CNCS, [Online]. Available: https://www.cncs.gov.pt/recursos/glossario/. [Acedido em 05 2018].

[44] NIST 800-61, “Computer Security Incident Handling Guide,” 2015.

[45] ISO/IEC 27035, “Information security incident management,” 2016. [Online]. Available: http://iso27001security.com/html/27035.html. [Acedido em 05 2018].

[46] ENISA, “Existing taxonomies,” [Online]. Available: https://www.enisa.europa.eu/topics/csirt-cert-services/community-projects/existing-taxonomies. [Acedido em 05 2018].

[47] Trustwave, “Global Security Report,” 2017.

[48] Symantec, “Internet Security Threat Report Volume 22,” 2017.

[49] Verizon, “Data Breach Investigations Report,” 2018.

[50] SecurityScorecard, “U.S. State and Federal Government Cybersecurity Report,” 2017.

[51] OWASP, “OWASP Top 10 Application Security Risks - 2017,” 2017. [Online]. Available: https://www.owasp.org/index.php/Top_10-2017_Top_10. [Acedido em 06 2018].

[52] Tenable, “Nessus - Vulnerability Scanner,” [Online]. Available: https://www.tenable.com/products/nessus/nessus-professional. [Acedido em 06 2018].

[53] “OpenVAS,” [Online]. Available: http://www.openvas.org/. [Acedido em 06 2018].

[54] Rapid7, “ nexpose - vulnerability scanner,” [Online]. Available: https://www.rapid7.com/products/nexpose/. [Acedido em 06 2018].

[55] EdgeScan, “EdgeScan,” [Online]. Available: https://www.edgescan.com/.

[56] netsparker, “Netsparker Web Application Security Scanner,” [Online]. Available: https://www.netsparker.com/. [Acedido em 06 2018].

[57] Acunetix, “Acunetix,” [Online]. Available: https://www.acunetix.com/. [Acedido em 06 2018].

[58] ISTQB, “What are the principles of testing?,” [Online]. Available: http://istqbexamcertification.com/what-are-the-principles-of-testing/. [Acedido em 07 2018].

[59] WPScan Team, [Online]. Available: https://wpscan.org/. [Acedido em 06 2018].

[60] C. Sullo e D. Lodge, “Nikto,” [Online]. Available: https://cirt.net/Nikto2.

[61] Greenbone, [Online]. Available: http://www.openvas.org/. [Acedido em 07 2018].

[62] Tenable, “Nessus Professional,” [Online]. Available: https://www.tenable.com/products/nessus/nessus-professional. [Acedido em 07 2018].

[63] ENISA, “Existing taxonomies,” [Online]. Available: https://www.enisa.europa.eu/topics/csirt-cert-services/community-projects/existing-taxonomies. [Acedido em 07 2018].

[64] C. Eiram e B. Martin, “The CVSSv2 Shortcomings, Faults, and Failures Formulation,” 2013.